Data Mining Lab, Exercise 4

Team# 1: Poliakov Valerii, Holovnia Dmytro, Selvaraj Sinju

Datasets: seeds.csv, seeds_description.txt

Tasks:

Carry out the cross-validation of the classification model provided by the k-nearest neighbors algorithm for data from file “seeds.csv”:

1. Twofold cross-validation: partition the data set into the training set and the test set. Compare the classifier results with the true values of the target variable. Present the confusion matrix and indicate the misclassified records. Provide the value of the error rate.
2. K-fold cross validation: set the number of folds K to 10. Provide an average value of the error rate as well as the values of the error rate and confusion matrices for all the folds. Perform calculations for two different number of neighbors in the knn algorithm, k = 1, and k >= 5. Comment the results.

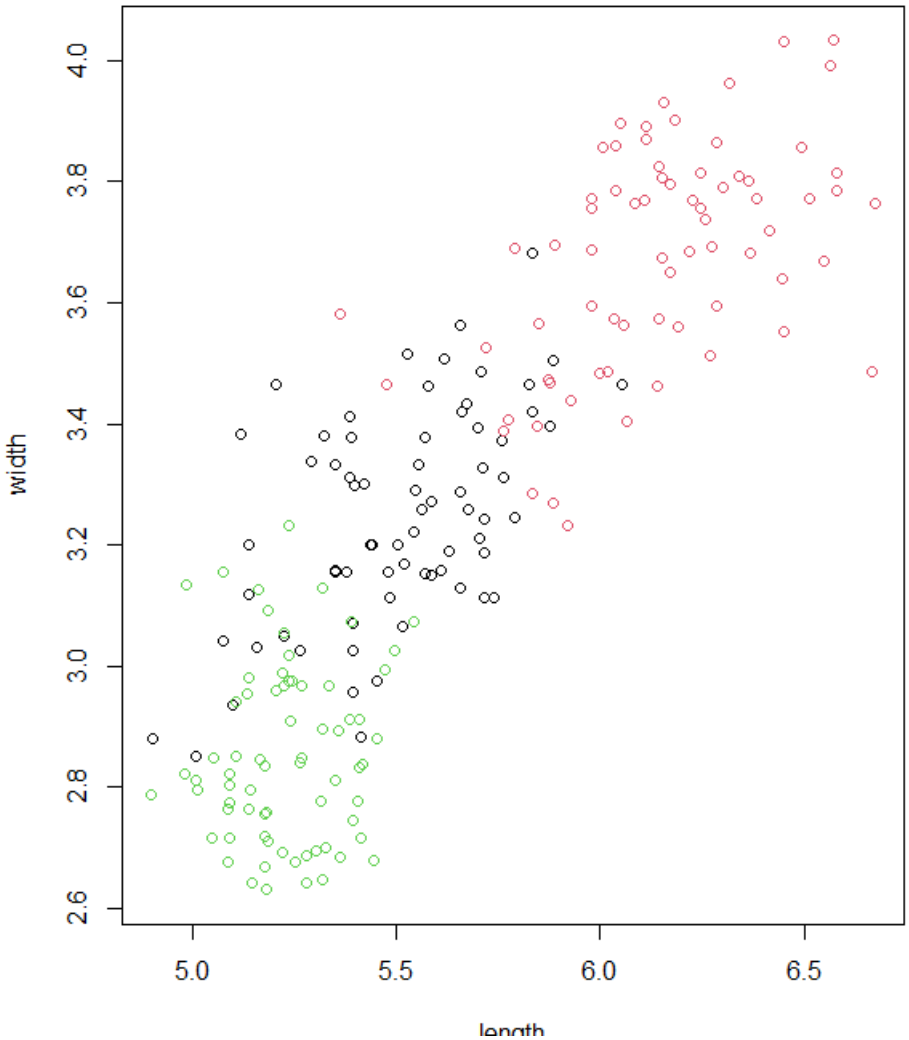
Task# 1 Twofold cross-validation

Step 1. Read the data. Read the seeds_description.txt to understand the data. We have wheat of 3 classes: Kama, Rosa and Canadian.

```
> library(class)
> d<-read.csv("seeds.csv", header=TRUE, sep=',')
> d
```

Step 2. Visualize class on scatter plot diagram using length and width attributes.

```
> with(d, plot(length, width, col =as.factor(class)))
```



Step 3. Data is ordered so we should randomly shuffle the data.

```
> d<-d[sample(nrow(d)),]
```

Step 4. Create test and training sets. First 20 records are the test dataset.

```
> test_indices<-c(1:20)  # first 20 records is the test dataset
> d.test <- d[test_indices,1:7]       #without wheat type (class)
> d.test.class<- d[test_indices,8]    #only wheat type (class)
> d.train <- d[-test_indices, 1:7]
> d.train.class <- d[-test_indices, 8]
```

Step 5. Run knn algorithm with 3 neighbors considered.

```
> knn.result <- knn(d.train, d.test, d.train.class, k = 3)
> knn.result
 [1] 1 3 3 3 2 2 3 3 2 2 3 3 1 1 2 2 2 2 3 2
Levels: 1 2 3
```

Step 6. Build confusion matrix and calculate error rate.

```
> t<-table(d.test.class,knn.result)
> err_rate<-mean(knn.result!=d.test.class)
>
> cat("\n\nerror rate = ",err_rate,"\n\n")


error rate =  0.15

> print(t)
             knn.result
d.test.class 1 2 3
           1 1 0 1
           2 0 9 0
           3 2 0 7
```

Step 7. Evaluate results.

Confusion matrix shows correct and incorrect classifications made by knn algorithm. Columns show prediction, rows show actual result.

- 2nd class identified correctly,
- two items of 3rd class incorrectly identified as 1st,
- one item of 1st class incorrectly identified as 3rd.

Error rate shows part of misclassified records from total number of records in the test set. In other words, we have 15% mistakes.

Task# 2 K-fold cross validation

Step 1. Create 10 equally sized folds and initialize the average error rate variable.

```
> nbreaks<-10
> folds <- cut(seq(1,nrow(d)),breaks=nbreaks,labels=FALSE)
> av_err_rate<-0
```

Step 2. Iterate over folds, use each fold as a test dataset and rest of the folds as a training subset. Combine the result of the models using averaging of voting. **1 nearest neighbors considered**.

```
> for(i in seq(1:nbreaks)){
+   test_indices <- which(folds==i,arr.ind=TRUE)
+   d.test <- d[test_indices,1:7]  #without class
+   d.test.class<- d[test_indices,8]    #only class
+   d.train <- d[-test_indices, 1:7]
```

```
+   d.train.class <- d[-test_indices, 8]
+
+   #Use the test and train data partitions in knn algorith
+   knn.result <- knn(d.train, d.test, d.train.class, k = 1)
+   t<-table(d.test.class,knn.result)
+   err_rate<-(nrow(d.test)-sum(diag(t)))/nrow(d.test)
+   av_err_rate = av_err_rate + err_rate
+
+   cat("\n\nFold = ",i,"\t error rate = ",err_rate,"\n")
+   print(t)
+ }


Fold =  1       error rate =  0.1428571
             knn.result
d.test.class 1 2 3
           1 1 0 1
           2 0 9 0
           3 2 0 8


Fold =  2       error rate =  0.1428571
             knn.result
d.test.class 1 2 3
           1 3 0 1
           2 2 6 0
           3 0 0 9


Fold =  3       error rate =  0.1428571
             knn.result
d.test.class 1 2 3
           1 6 1 2
           2 0 5 0
           3 0 0 7


Fold =  4       error rate =  0.0952381
             knn.result
d.test.class 1 2 3
           1 8 0 2
           2 0 7 0
           3 0 0 4


Fold =  5       error rate =  0.0952381
             knn.result
d.test.class 1 2 3
           1 9 1 1
           2 0 3 0
           3 0 0 7


Fold =  6       error rate =  0.04761905
             knn.result
d.test.class  1  2  3
           1  5  0  0
           2  0 12  0
           3  1  0  3


Fold =  7       error rate =  0.0952381
             knn.result
d.test.class  1  2  3
           1  3  1  0
           2  1  5  0
           3  0  0 11


Fold =  8       error rate =  0.0952381
             knn.result
d.test.class 1 2 3
           1 8 0 1
           2 1 4 0
           3 0 0 7
```

```
Fold =  9       error rate =  0
            knn.result
d.test.class 1 2 3
           1 7 0 0
           2 0 8 0
           3 0 0 6


Fold =  10      error rate =  0.0952381
            knn.result
d.test.class 1 2 3
           1 8 1 0
           2 1 6 0
           3 0 0 5
> av_err_rate = av_err_rate/nbreaks
> av_err_rate
[1] 0.2142857
```

The average error rate is 0,2142857 for the number of neighbors = 1.

Step 3. Repeat K-fold cross validation for the number of neighbors = 7

```
> for(i in seq(1:nbreaks)){
+   test_indices <- which(folds==i,arr.ind=TRUE)
+   d.test <- d[test_indices,1:7]  #without class
+   d.test.class<- d[test_indices,8]    #only class
+   d.train <- d[-test_indices, 1:7]
+   d.train.class <- d[-test_indices, 8]
+
+   #Use the test and train data partitions in knn algorith
+   knn.result <- knn(d.train, d.test, d.train.class, k = 7)
+   t<-table(d.test.class,knn.result)
+   err_rate<-(nrow(d.test)-sum(diag(t)))/nrow(d.test)
+   av_err_rate = av_err_rate + err_rate
+
+   cat("\n\nFold = ",i,"\t error rate = ",err_rate,"\n")
+   print(t)
+ }


Fold =  1       error rate =  0.0952381
            knn.result
d.test.class 1 2 3
           1 2 0 0
           2 0 9 0
           3 2 0 8


Fold =  2       error rate =  0.1428571
            knn.result
d.test.class 1 2 3
           1 3 1 0
           2 1 7 0
           3 1 0 8


Fold =  3       error rate =  0.1428571
            knn.result
d.test.class 1 2 3
           1 6 1 2
           2 0 5 0
           3 0 0 7


Fold =  4       error rate =  0.1904762
            knn.result
d.test.class 1 2 3
           1 6 0 4
           2 0 7 0
           3 0 0 4


Fold =  5       error rate =  0.04761905
```

```
            knn.result
d.test.class  1  2  3
           1 10  1  0
           2  0  3  0
           3  0  0  7


Fold =  6       error rate =  0.1428571
            knn.result
d.test.class  1  2  3
           1  4  0  1
           2  1 11  0
           3  1  0  3


Fold =  7       error rate =  0.0952381
            knn.result
d.test.class  1  2  3
           1  4  0  0
           2  2  4  0
           3  0  0 11


Fold =  8       error rate =  0.04761905
            knn.result
d.test.class 1 2 3
           1 8 0 1
           2 0 5 0
           3 0 0 7


Fold =  9       error rate =  0
            knn.result
d.test.class 1 2 3
           1 7 0 0
           2 0 8 0
           3 0 0 6


Fold =  10      error rate =  0.0952381
            knn.result
d.test.class 1 2 3
           1 7 1 1
           2 0 7 0
           3 0 0 5
> av_err_rate = av_err_rate/nbreaks
> av_err_rate
[1] 0.1214286
```

The average error rate is 0,1214286 for the number of neighbors = 7.

The more neighbors we use the better prediction result we have. For example, for 30 neighbors the error rate is 0,102619